

MÓDULO 2: SINTAXIS BASICA

Eduard Lara

INDICE

1. Sintaxis básica de PHP
2. Constantes y variables
3. Ambito de variables
4. Arrays
5. Operadores

1. SINTAXIS BASICA DE PHP

- ❖ ¿Cómo identifica el servidor web que un documento contiene elementos php? El documento deberá llevar la extensión php.
- ❖ ¿Cómo detecta el interprete de php fragmentos de código php?
¿Cómo se incrusta en la página web?

```
<?php  
    instrucciones_en_codigo_php;  
    <?php echo "$variable"; ?>  
?>
```

```
<?= expresion ?> //equivale a <? echo expresion; ?>  
<?= $variable ?>
```

1. SINTAXIS BASICA DE PHP

Reglas básicas para las instrucciones:

- ❖ PHP es sensible a las mayúsculas
- ❖ Las instrucciones deben acabar en punto y coma (;) como en C.
La marca final ?> implica un ;
- ❖ Los comentarios son iguales que en Javascript ó C:
 - //→ Para comentar una línea (estilo C++ y Java)
`print "hola"; // Comentario de una línea`
 - /* y */ → Para comentar más de una línea (estilo C)
`/* Comentario de
varias líneas */`
 - # → Comentarios estilo shell

1. SINTAXIS BASICA DE PHP

Comparativa entre PHP, JSP Y ASP

PHP	JSP	ASP
<?php	<%	<%
?>	%>	%>
Apache	Tomcat	IIS

1. SINTAXIS BASICA DE PHP

ECHO Y PRINT

Echo y Print se utilizan enviar datos al navegador web del cliente.

- ❖ echo: muestra una o más cadenas

```
echo cadena1 [, cadena2...]
```

```
echo "Hola mundo";
```

```
echo "Hola ", "mundo";
```

- ❖ print: muestra una cadena

```
print cadena;
```

```
print "Hola mundo";
```

```
print "Hola "."mundo"; //Operador de concatenación
```

1. SINTAXIS BASICA DE PHP

PRIMER EJEMPLO

```
<html>
<head> <title>Mi primer programa en PHP</title>
</head>
<body>
  <?php
    echo "<h1>HOLA MUNDO</h1>";
    print ("<P>Hola mundo</P>");
  ?>
</body>
</html>
```

Guardar en c:\xampp\htdocs o subdirectorio.
Para probarlo, escribir en un navegador:
<http://localhost/primer.php>

1. SINTAXIS BASICA DE PHP

USO DE \n PARA GENERAR CÓDIGO HTML LEGIBLE

	Sin \n	Con \n
Código PHP	<pre>print("<P>Párrafo 1</P>"); print("<P>Párrafo 2</P>");</pre>	<pre>print("<P>Párrafo 1</P>\n"); print("<P>Párrafo 2</P>\n");</pre>
Código HTML	<pre><P>Párrafo 1</P><P>Párrafo2</P></pre>	<pre><P>Párrafo 1</P> <P>Párrafo 2</P></pre>
Salida	<pre>Párrafo 1 Párrafo 2</pre>	<pre>Párrafo 1 Párrafo 2</pre>

\n es útil para hacer código HTML legible, pero para hacer un salto de línea en HTML se debe de poner

2. VARIABLES

PHP soporta 8 tipos de datos primitivos:

- ⦿ Tipos escalares: boolean, integer, float (also called double), string
- ⦿ Tipos compuestos: array, object
- ⦿ Tipos especiales: resource, NULL
- ⦿ Tipo **integer** (números enteros)
 - ⦿ 27, -5, 0
- ⦿ Tipo **double** (números reales)
 - ⦿ 1.234, -5.33
- ⦿ Tipo **boolean** (lógico)
 - ⦿ Valores: *true*, *false* (insensibles a las mayúsculas)
 - ⦿ El 0 y la cadena vacía tienen valor *false*

```
<?php  
$txt = "Hello world!";  
$x = 5;  
$y = 10.5;  
?>
```

2. VARIABLES

- ❖ Todos los nombres de las variables siempre van precedidas del símbolo dólar (\$):
`$Mivariable=valor;`
`$edat = 7;`
`$nom="Xavier";`
- ❖ Las variables se crean cuando se utilizan.
- ❖ No hace falta declararlas ni especificar el tipo de una variable
- ❖ La variable tomará el tipo de dato que contenga el valor que se le asigne.
- ❖ El nombre es sensible a las mayúsculas: `$Foo != $foo != $fOo`
- ❖ Los nombres de las variables no pueden empezar con números ni caracteres especiales. Comienzan por letra o subrayado, seguido de letras, números o subrayado.

2. VARIABLES

```
1 <html>
2 <head>
3   <title>Problema</title>
4 </head>
5 <body>
6 <?php
7     $dia = 24; //Se declara una variable de tipo integer.
8     $sueldo = 758.43; //Se declara una variable de tipo double.
9     $nombre = "juan"; //Se declara una variable de tipo string.
10    $exite = true; //Se declara una variable boolean.
11    echo "Variable entera:";
12    echo $dia;
13    echo "<br>";
14    echo "Variable double:";
15    echo $sueldo;
16    echo "<br>";
17    echo "Variable string:";
18    echo $nombre;
19    echo "<br>";
20    echo "Variable boolean:";
21    echo $exite;
22 >?>
23 </body>
24 </html>
```

2. VARIABLES

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
    <p>
      <?php
        $productes=2;
        $preu_producte=100;
        echo "el preu total:". $productes*$preu_producte;
      ?>
    </p>
  </body>
</html>
```

```
<html>
  <head>
    <title>PHP!</title>
    <meta charset="utf-8">
  </head>
  <body>
    <?php
      $nom="Pepet";
      $edat=30;
      echo $nom." té ". $edat." \n";
      echo "el proper any tindrà " . ($edat+1) . " anys";
    ?>
  </body>
```

2. VARIABLES

- ❖ Aunque no se define el tipo de variable, se puede forzar con settype:

```
<?php
    $var1 = "3";
    settype ( $var1, 'integer' );//Forzamos cambio a entero
    echo $var1; // 3
?>
```

- ❖ Ciertos nombres de variables están reservadas en PHP
 - ⦿ Variables de formularios (\$_POST, \$_GET)
 - ⦿ Variables de servidor (\$_SERVER)

2. VARIABLES

TIPO STRING

Las cadenas se encierran entre comillas simples o dobles:

- ‘simples’: admite los caracteres de escape \’ (comilla simple) y \\ (barra). Las variables **NO** se expanden
- “dobles”: admite más caracteres de escape, como \n, \r, \t, \\, \\$, \". Los nombres de variables **SÍ** se expanden:


```
$a = 9;  
print 'a vale $a\n'; // muestra a vale $a\n  
print "a vale $a\n"; // muestra a vale 9 y avanza una línea  
print "<IMG SRC='logo.gif'>"; // muestra <IMG SRC='logo.gif'>  
print "<IMG SRC=\"logo.gif\">"; // muestra <IMG SRC=\"logo.gif\">
```

- Acceso a un carácter de la cadena: La forma es \$inicial = \$nombre{0};

2. VARIABLES

TIPO STRING

```
<?php
    $foo = 25;           // Variable numérica
    $bar = "Hello";      // String
    $foo = ($foo * 7);    // Multiplica foo por 7
    $bar = ($bar * 7);    // Expresión inválida
    $foo = 25;           // Variable numérica
    $bar = "Hello";      // String
    echo $bar;           // Muestra Hello
    echo $foo, $bar;      // Muestra 25Hello
    echo "5x5=", $foo;    // Muestra 5x5=25
    echo "5x5=$foo";      // Muestra 5x5=25
    echo '5x5=$foo';      // Muestra 5x5=$foo
?>
```



Los strings en comillas simples (' ') no son interpretados o evaluados por PHP. Esto pasa con variables y secuencias de escape tales como "\n" or "\\"

2. VARIABLES

TIPO STRING

- ❖ Para concatenar un variables tipo string se usa el Operador “.” (Concatena):

```
2  <html>
3  <head>
4    <title>Problema</title>
5  </head>
6  <body>
7
8    <?php
9      $cadena1="diego";
10     $cadena2="juan";
11     $cadena3="ana";
12     $todo=$cadena1.$cadena2.$cadena3."<br>";
13     echo $todo;
14     $edad1=24;
15     echo $cadena1." tiene $edad1 de edad";
16   ?>
17 </body>
18 </html>
```

- ❖ La conversión del valor de \$entera a una cadena de texto se realiza mediante la utilización del operador (.).

**echo "La variable 'entera'
vale".\$entera.".";**

2. VARIABLES

TIPO STRING. Carácter de escape

Si el string tiene un conjunto de comillas dobles que deben de ser visibles, usa la contrabarra \ [backslash] antes de la comilla para mostrarla

```
<?php  
$heading="\\"Computer Science\\"";  
Print $heading;  
?>
```

```
"Computer Science"
```

2. VARIABLES

VARIABLE DE VARIABLES

- ⦿ Se pueden crear nombres de variables dinámicamente
- ⦿ La variable de variable toma su nombre del valor de otra variable previamente declarada

```
$a = "hola";  
$$a = "mundo";  
print "$a $hola\n";  
print "$a ${$a}";  
Resultado:  
    hola mundo  
    hola mundo
```

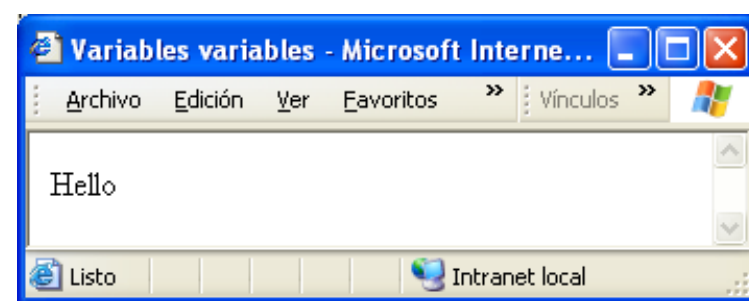
2. VARIABLES

VARIABLE DE VARIABLES: Página internacionalizada

```
<?PHP
    $mensaje_es="Hola";
    $mensaje_en="Hello";
    $idioma = "es";
    $mensaje = "mensaje_" .
$idioma;
    print $$mensaje;
?>
```



```
<?PHP
    $mensaje_es="Hola";
    $mensaje_en="Hello";
    $idioma = "en";
    $mensaje = "mensaje_" .
$idioma;
    print $$mensaje;
?>
```



2. VARIABLES

FUNCIONES

❖ Funciones para realizar una conversión de tipos:

- ◉ doubleval (cadena a real)
- ◉ intval (cadena a entero)
- ◉ strval (número a cadena)

```
$Micad = "1235";  
$MiReal = intval($Micad);
```

❖ Otras funciones útiles son:

- ◉ isset(variable); // comprueba si existe una variable
- ◉ empty(variable); // comprueba si una variable tiene un valor asignado.
- ◉ settype(variable, tipo); // asigna un tipo de dato a una var
- ◉ gettype(variable); // devuelve el tipo de dato de una var
- ◉ unset(variable); // elimina variables

2. VARIABLES

❖ Funciones de interés:

- ⊙ Las funciones *is_type* comprueban si una variable es de un tipo dado: `is_array()`, `is_bool()`, `is_float()`, `is_integer()`, `is_null()`, `is_numeric()`, `is_object()`, `is_resource()`, `is_scalar()`, `is_string()`
- ⊙ La función `var_dump()` muestra el tipo y el valor de una variable. Es especialmente interesante con los arrays

```
<?php  
echo strpos("Hello world!", "world"); // outputs 6  
?>
```

```
<?php  
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!  
?>
```

2. VARIABLES

```
<?php  
echo strlen("Hello world!"); // outputs 12  
?>
```

```
<?php  
echo str_word_count("Hello world!"); // outputs 2  
?>
```

```
<?php  
echo strrev("Hello world!"); // outputs !dlrow olleH  
?>
```

2. VARIABLES

CONSTANTES EN PHP

- ❖ Las constantes se escriben sin el \$ y en MAYUSCULAS. Se definen así:
`define("Nombre_constante","Valor");`
`define("PI",3.1515926);`
`echo PI; //`No llevan \$ delante
- ❖ Sus valores se mantienen en todo el documento.
- ❖ Sólo se pueden definir constantes de los tipos escalares (boolean, integer, double, string)
- ❖ Para comprobar si la constante existe o esta definida, se utiliza la función:
`defined(constante);`
`defined("PI");` → devuelve verdadero si existe

2. VARIABLES

CONSTANTES EN PHP

```
<html>
  <head><title>PHP!</title>  <meta charset="utf-8"></head>
  <body>
    <?php
      define("IVA", 21.0/100.0);
      $preu_sense_iva=100.0;
      echo "Preu amb IVA ", ":", $preu_sense_iva+$preu_sense_iva*IVA;
    ?>
  </body>
```


3. AMBITO VARIABLES

El **ámbito de una variable** es la porción de código del programa desde el cual es accesible.

- ❖ **Variables locales** → Variables definidas dentro de un script o función y sólo pueden ser accedidas dentro de ese ámbito.
- ❖ **Variables globales** → Variables que pueden ser usadas desde cualquier lugar, por ejemplo desde una función, aunque estén definidas fuera de ella. Su sintaxis es: `global "variable";`
- ❖ **Variables superglobales** → Variables predefinidas, que se comportan como si fueran globales: `$_SERVER, $_GLOBALS, $_SERVER, $_GET, $_POST, $_ENV, $_COOKIES, $_FILES, $_REQUEST, $_SESSION`

3. AMBITO VARIABLES

AMBITO GLOBAL

```
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}

myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

AMBITO LOCAL

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}

myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

3. AMBITO VARIABLES

VARIABLES GLOBALES

```
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```

3. AMBITO VARIABLES

VARIABLES GLOBALES

```
<?php
$x = 5;
$y = 10;

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

3. AMBITO VARIABLES

VARIABLES ESTATICAS

```
<!DOCTYPE html>
<html>
<body>

<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>

</body>
</html>
```

0
1
2

4. ARRAYS

- ❖ Los arrays son un conjunto de valores almacenados en una sola variable
- ❖ Es un tipo de dato estructurado que permite almacenar un conjunto de dato homogéneo, es decir, donde todos son del mismo tipo
- ❖ Son como tablas, donde los valores de sus celdas son accesibles mediante índices (enteros no negativos o claves).
- ❖ Los elementos que la componen se identifican mediante el nombre del array y un índice:

`$nombre[indice] ;`

4. ARRAYS

Hay 3 maneras de indexar(referirnos) a los diferentes elementos de un array:

- Arrays escalares (indexados numericamente): Si nos referimos mediante un número (que empezará por 0).
- Arrays asociativos (indexados por clave): Si existe un mapeo clave(palabra)- valor.
- Multidimensional: Si en cada índice, hay otro array.

Indexación	Construcción
Arrays escalares	Estáticos
Arrays asociativos	Dinámicos

4. ARRAYS

CONSTRUCCIÓN ESTÁTICA

- ❖ Un array se puede construir de forma estática mediante la instrucción array:

```
Array = array([index]=>[valor], [index2]=>[valor], ...);  
$ciudades = array(0=>"madrid", 1=>"barcelona", 2=>"lugo");  
//Coinciden ambas declaraciones  
$ciudades = array("madrid", "barcelona", "lugo");  
  
echo $ciudades[0]; //Imprime "madrid"  
echo $ciudades[1]; //Imprime "barcelona"  
echo $ciudades[2]; //Imprime "lugo"
```


4. ARRAYS

CONSTRUCCIÓN DINAMICA

❖ Se puede insertar nuevos valores a un array dinámicamente, posteriormente a su declaración:

```
$ciudades = array("madrid", "barcelona", "lugo");  
$ciudades[3] = "bilbao";  
$ciudades[4] = "leon";  
$ciudades[5] = "segovia";
```

```
echo $ciudades[2]; //Imprime "lugo"  
echo $ciudades[3]; //Imprime "bilbao"  
echo $ciudades[4]; //Imprime "leon"
```

4. ARRAYS

ARRAYS ESCALARES (INDEXADOS NUMERICAMENTE)

- ❖ Se indexan mediante claves o índices numéricos (enteros no negativos), donde el índice del primer elemento es el 0.

```
$dias_semana=array('Lunes','Martes','Miercoles','Jueves','Viernes');  
echo $dias_semana[3]." ".$dias_semana[0];
```

→ Imprime Jueves Lunes

```
$medidas=array(10, 25, 15);  
echo $medidas[0]." ".$medidas[2];
```

→ Imprime 10 15

4. ARRAYS

ARRAYS ESCALARES (INDEXADOS NUMERICAMENTE)

```
$cars = array("Volvo", "BMW", "Toyota");
```

```
$cars[0] = "Volvo";  
$cars[1] = "BMW";  
$cars[2] = "Toyota";
```

```
<?php  
$cars = array("Volvo", "BMW", "Toyota");  
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";  
?>
```

4. ARRAYS

ARRAYS ASOCIATIVOS

- ❖ Array indexado por una clave de tipo cadena.
- ❖ Está formado por conjuntos de parejas índice => valor, donde los índices son strings o cadenas de texto.
- ❖ Acceso a los elementos de un array asociativo:

`$nombre['identificador'] = valor;`

```
$coche = array('marca'=>'mercedes', 'matricula'=>'4567',  
'modelo'=>'clase-E', 'color'=>'negro');
```

```
echo $coche['marca']; → Imprime mercedes
```

```
$color = array ('rojo'=>101, 'verde'=>51, 'azul'=>255);
```

```
Echo $color['rojo'] → Imprime 101
```

4. ARRAYS

ARRAYS ASOCIATIVOS

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

```
$age['Peter'] = "35";  
$age['Ben'] = "37";  
$age['Joe'] = "43";
```

```
<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
echo "Peter is " . $age['Peter'] . " years old."  
?>
```

4. ARRAYS

ARRAYS ASOCIATIVOS

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

4. ARRAYS

AÑADIR ELEMENTOS A UN ARRAY

```
echo "Añadimos el color violeta<br>";
```

```
$colors[]="violeta";
```

```
echo "Hay ".count($colors)." elementos, que son: ";
```

```
for($i=0;$i<count($colors) + 1;$i++)
```

```
    echo $colors[$i].", ";
```

```
→ Hay 5 elementos, que son: rojo, , azul, rosa, amarillo, violeta,
```

```
echo "Añadimos el color marron<br>";
```

```
$colors[1]="marron";
```

```
echo "Hay ".count($colors)." elementos, que son: ";
```

```
for($i=0;$i<count($colors) ;$i++)
```

```
    echo $colors[$i].", ";
```

```
?> → Hay 6 elementos, que son: rojo, marron, azul, rosa, amarillo, violeta,
```

Cuando insertemos un elemento sin indicar ningún índice, PHP lo añade al final.

Insertamos un elemento en el espacio anterior verde

4. ARRAYS

ELIMINAR ELEMENTOS DE UN ARRAY

```
<?php
```

```
$colors=array("rojo","verde","azul","rosa","amarillo");
```

```
echo "Hay ".count($colors)." elementos, que son: ";
```

```
for ($i=0;$i<count($colors);$i++)
```

```
    echo $colors[$i].", ";
```

→ Hay 5 elementos, que son: rojo, verde, azul, rosa, amarillo,

```
echo "Eliminamos el color verde<br>";
```

```
unset($colors[1]);
```



```
echo "Hay ".count($colors)." elementos, que son: ";
```

```
for($i=0;$i<count($colors) + 1; $i++)
```

```
    echo $colors[$i].", ";
```

→ Hay 4 elementos, que son: rojo, , azul, rosa, amarillo,

La función "unset()" vacía el elemento del array (el número de elementos de este se decrementa en uno), pero no se corren todos los elementos.

4. ARRAYS

ARRAYS: FOR VS FOREACH

El hecho de eliminar un elemento y que su posición no se rellene provoca un problema en los contadores de los for's, ya que no hay ningún valor en el índice eliminado. La instrucción foreach muestra todos los elementos que hay en un array y que contengan información, despreciando el resto.

```
<?php
```

```
$colors=array("rojo","verde","azul","rosa","amarillo");  
echo "Hay ".count($colors)." elementos, que son: ";  
foreach($colors as $val) echo $val.",";
```

→ Hay 5 elementos, que son: rojo, verde, azul, rosa, amarillo,

```
echo "<br>Eliminamos el color verde<br>";
```

```
unset($colors[1]);
```

```
echo "Hay ".count($colors)." elementos, que son: ";
```

```
foreach($colors as $val) echo $val.",";
```

→ Hay 4 elementos, que son: rojo, azul, rosa, amarillo,

```
?>
```

```
foreach(array as $clave => $valor)
```

```
foreach(array as $valor)
```

```
foreach($mi_agenda as $nombre => $mail)
```

```
echo "El mail de ".$clave. "es ". $value . "<br>";
```

4. ARRAYS

ARRAYS MULTIDIMENSIONALES

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

```
$cars = array  
(  
    array("Volvo",22,18),  
    array("BMW",15,13),  
    array("Saab",5,2),  
    array("Land Rover",17,15)  
);
```

4. ARRAYS

ARRAYS MULTIDIMENSIONALES

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array
(
    array("Volvo",22,18),
    array("BMW",15,13),
    array("Saab",5,2),
    array("Land Rover",17,15)
);

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>

</body>
</html>
```

Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

5. OPERADORES EN PHP7

- Operadors aritmètics
- Operadors d'assignació
- Operadors de comparació
- Operadors d'increment / decreixement
- Operadors lògics
- Operadors de *strings*
- Operadors d'*arrays*
- Operadors d'assignació condicional

5. OPERADORES EN PHP7

Aritméticos

Operator	Name	Example
+	Addition	$\$x + \y
-	Subtraction	$\$x - \y
*	Multiplication	$\$x * \y
/	Division	$\$x / \y
%	Modulus	$\$x \% \y
**	Exponentiation	$\$x ** \y

5. OPERADORES EN PHP7

Asignacion

Assignment	Same as...	Description
<code>x = y</code>	<code>x = y</code>	The left operand gets set to the value of the expression on the right
<code>x += y</code>	<code>x = x + y</code>	Addition
<code>x -= y</code>	<code>x = x - y</code>	Subtraction
<code>x *= y</code>	<code>x = x * y</code>	Multiplication
<code>x /= y</code>	<code>x = x / y</code>	Division
<code>x %= y</code>	<code>x = x % y</code>	Modulus

5. OPERADORES EN PHP7

Comparación

Operator	Name	Example
==	Equal	\$x == \$y
===	Identical	\$x === \$y
!=	Not equal	\$x != \$y
<>	Not equal	\$x <> \$y
!==	Not identical	\$x !== \$y
>	Greater than	\$x > \$y
<	Less than	\$x < \$y
>=	Greater than or equal to	\$x >= \$y
<=	Less than or equal to	\$x <= \$y
<=>	Spaceship	\$x <=> \$y

5. OPERADORES EN PHP7

Incremento/decremento

Els operadors d'increment de PHP s'utilitzen per incrementar el valor d'una variable.

Els operadors de decreixement de PHP s'utilitzen per disminuir el valor d'una variable.

Operator	Name	Description
<code>++\$x</code>	Pre-increment	Increments <code>\$x</code> by one, then returns <code>\$x</code>
<code>\$x++</code>	Post-increment	Returns <code>\$x</code> , then increments <code>\$x</code> by one
<code>--\$x</code>	Pre-decrement	Decrements <code>\$x</code> by one, then returns <code>\$x</code>
<code>\$x--</code>	Post-decrement	Returns <code>\$x</code> , then decrements <code>\$x</code> by one

5. OPERADORES EN PHP7

Lógicos

Els operadores lògics de PHP s'utilitzen per combinar sentències condicionals.

Operator	Name	Example	Result
and	And	<code>\$x and \$y</code>	True if both <code>\$x</code> and <code>\$y</code> are true
or	Or	<code>\$x or \$y</code>	True if either <code>\$x</code> or <code>\$y</code> is true
xor	Xor	<code>\$x xor \$y</code>	True if either <code>\$x</code> or <code>\$y</code> is true, but not both
<code>&&</code>	And	<code>\$x && \$y</code>	True if both <code>\$x</code> and <code>\$y</code> are true
<code> </code>	Or	<code>\$x \$y</code>	True if either <code>\$x</code> or <code>\$y</code> is true
<code>!</code>	Not	<code>!\$x</code>	True if <code>\$x</code> is not true

- Operador de control de error
@. Antepuesto a una expresión, evita cualquier mensaje de error que pueda ser generado por la expresión

5. OPERADORES EN PHP7

- Precedencia de operadores (de mayor a menor):

++, --
*, /, %
+, -
<, <=, >, >=
==, !=
&&, and
, or

5. OPERADORES EN PHP7

Strings

PHP té dos operadores que estan especialmente diseñados para cadenas.

Operator	Name	Example	Result
.	Concatenation	<code>\$txt1 . \$txt2</code>	Concatenation of <code>\$txt1</code> and <code>\$txt2</code>
<code>.=</code>	Concatenation assignment	<code>\$txt1 .= \$txt2</code>	Appends <code>\$txt2</code> to <code>\$txt1</code>

5. OPERADORES EN PHP7

Arrays

Els operadors d'*array* de PHP s'utilitzen per comparar *arrays*.

Operator	Name	Example	Result
+	Union	<code>\$x + \$y</code>	Union of <code>\$x</code> and <code>\$y</code>
<code>==</code>	Equality	<code>\$x == \$y</code>	Returns true if <code>\$x</code> and <code>\$y</code> have the same key/value pairs
<code>===</code>	Identity	<code>\$x === \$y</code>	Returns true if <code>\$x</code> and <code>\$y</code> have the same key/value pairs in the same order and of the same types
<code>!=</code>	Inequality	<code>\$x != \$y</code>	Returns true if <code>\$x</code> is not equal to <code>\$y</code>
<code><></code>	Inequality	<code>\$x <> \$y</code>	Returns true if <code>\$x</code> is not equal to <code>\$y</code>
<code>!==</code>	Non-identity	<code>\$x !== \$y</code>	Returns true if <code>\$x</code> is not identical to <code>\$y</code>

6. REPORTANDO ERRORES

- ❖ Cuando se produce un error en la ejecución de un programa en PHP, este puede llegar a provocar l'aturada del programa. Además, el error puede mostrarse en pantalla con más o menos grado de detalle.

 Parse error: syntax error, unexpected '\$float' (T_VARIABLE) in C:\xampp\htdocs\PhpProject1\index.php on line 16

- ❖ Lo más normal en entornos de desarrollo es que nos muestre explícitamente el error al ejecutar el programa tal y como se ve en la imagen pero...

6. REPORTANDO ERRORES

- ❖ Que hacemos si esto nos pasa en un servidor externo y no podemos ver los errores por pantalla? Por defecto, no se muestran los mensajes de error en PHP.
- ❖ Pues que habilitamos la muestra de errores explicitándolo en el mismo código fuente.

```
<?php
    ini_set('display_errors',0);
    ini_set('display_startup_errors',0);
    error_reporting(E_ALL);
```

Existen más niveles de error (E_ALL, E_ERROR, E_CORE_ERROR...) pero con este parámetro se mostraran todos

Tambien se puede editar php.ini.

- 1) Cambiar :
error_reporting = E_ALL & ~E_DEPRECATED por:
error_reporting = E_ALL
- 2) Cambiar:
display_errors = Off por:
display_errors = On
- 3) Reiniciar el servidor